Ziegenhagen, Uwe; Klinke, Sigbert; Härdle, Wolfgang Karl

Working Paper

Yxilon: Designing The Next Generation, Vertically Integrable Statistical Software Environment

Papers / Humboldt-Universität Berlin, Center for Applied Statistics and Economics (CASE), No. 2004,40

Provided in cooperation with:
Humboldt-Universität Berlin

Suggested citation: Ziegenhagen, Uwe; Klinke, Sigbert; Härdle, Wolfgang Karl (2004) : Yxilon: Designing The Next Generation, Vertically Integrable Statistical Software Environment, Papers / Humboldt-Universität Berlin, Center for Applied Statistics and Economics (CASE), No. 2004,40, http://hdl.handle.net/10419/22213

Yxilon – Designing The Next Generation, Vertically Integrable Statistical Software Environment*

Wolfgang Härdle[1,2], Sigbert Klinke[2] and Uwe Ziegenhagen[1,2]
Humboldt-Universität zu Berlin
[1]Center for Applied Statistics and Economics
[2]Institute for Statistics and Econometrics
{haerdle, sigbert, ziegenhagen}@wiwi.hu-berlin.de

July 31, 2004

Abstract

Modern statistical computing requires smooth integration of new algorithms and quantitative analysis results in all sorts of platforms such as webbrowsers, standard and proprietary application software. Common statistical software packages can often not be adapted to integrate into new environments or simply lack the demands users and especially beginners have.

With Yxilon we propose a vertically integrable, modular statistical computing environment, providing the user a rich set of methods and a diversity of different interfaces, including command-line interface, web clients and interactive examples in electronic books. This architecture allows the users to rely upon only one environment in order to organize data from a variety of sources, analyse them and visualize or export the results to other software programs.

The design of Yxilon is inspired by XploRe, a statistical environment developed by MD*Tech and Humboldt-Universität zu Berlin. Yxilon incorporates several ideas from recent developments and design principles in software engineering: modular plug-in architecture, platform independence, and separation of user interfaces and computing engine.

Keywords Java, Client/Server, XploRe, Yxilon, electronic publishing, e-books

*Financial support was received from Society for Economics and Management at Humboldt-Universität zu Berlin.

1 The Past and Presence of Statistical Software

> "Each new generation of computers offers us new possiblities, at a time when we are far from using most of the possibilities offered by those already obsolete."
>
> John W. Tukey (1965)

Before we discuss the future of statistical software it is worth to look at past and present tools for a statistican's daily work. Back in the 60s of the last century, where several commercial software packages originally date from, data analysis was an uphill struggle. Computers were scarce, expensive and, compared with the latest generation one can buy in each supermarket today, extremely slow. Furthermore these machines had to be operated by specially trained personnel and computing time even had to be reserved in advance.

When we look at the market for statistics software today, we find a large variety of free or commercial packages as S-Plus, SPSS, Minitab and R, being able to handle large data in a split second. But really astonishing is that the product used for most analysis tasks worldwide is Microsoft Excel, although there are certain doubts about its numerial precision (McCullough and Wilson, 1999).

Why is Excel thus attractive for so many users? This must have to do with the way we do data analysis. Following Chambers (2000) we can split this job into three different subtasks: Organisation, Analysis and Presentation. In this sequence data analysis is done today. Table 1 points out the pros and cons in these three subtasks. We see, Excel may be a good choice for small or medium tasks, but may not be sufficient for a deeper analysis of large datasets. Here "dedicated" statistics packages find their market. The ability to compete against a standard spreadsheet package may be taken as one requirement but what other requirements does a statistics package need to fulfill?

	pro	contra
Organisation	large variety of import & exportfilters, filter and database functions	tables limited to 65.536 rows and 256 columns
Analysis	numerous functions for statistical analysis, e.g. for ANOVA, Fourier Analysis, Regression and Sampling	numerical inaccuracies
Presentation	graphics connected with data dynamically	no statistical displays as boxplots, histograms, etc.

Table 1: Performance of Excel in Organisation, Analysis and Presentation

This function is discussed in the next section, where we set up a list of 12 essentials for an universal, next generation statistical computing environment. In Section 3 we argue that vertical integration is the key for successful data analysis, proloferation of methods and report generation. In Section 4 we sketch the design of Yxilon (*Y*es, *X*ploRe *I*s *L*iving *On*), a prototype for a future, vertically integrated statistical computing environment.

In the last section we summarize our thoughts. All information may also be found on the projects webpage http://www.quantlet.org.

2 Requirements for Statistical Environments

Chambers and Lang (1999) give a list of essentials that are needed and desirable for statistical tools:

1. Usable from multiple front-ends, i.e. the data analysis must be performed from Excel, Web-browsers and other user interfaces.
2. Support for development of GUIs for different audiences. Each discipline has its own culture of naming statistical phenomena, GUIs must reflect this feature.
3. Extensibilty on language/interpreter level and native core level, i.e. high level code must be converted to high speed production code.
4. Internet abilities to read and write data to networks
5. Database support to allow nearly real-time analysis
6. Interactive graphics and connection to other graphical controls. Teachers and students like to show and study the sensitivity of the implemented statistical methods.
7. Support for multi-processor machines
8. Extensibility by inclusion of existing code
9. Optimization for performance

The view of an econometrician on methods and data differs from the view a biometrician has although they might use exactly the same statistical techniques. A Japanese statistician has different needs for supporting help system than say a French statistician, thus there is a need for more than one language interface. Reports, tutorials and newsgroups are ressources that influence the choice of the statistical environment.

We therefore add several essentials:

Set of methods: The included set of methods is for sure one of the most crucial points when selecting a statistical engine. Although there is a common subset of methods, most commercial software programs show huge differences in the included method sets, most producers provide (expensive) add-ons for special analysis tasks.

Multiple language support: English is the lingua franca of science but for non-native users the *usability* of the software increases significantly when graphical user interface, hints and especially error messages are given in their own tongue.

Valuable user ressources: The available user ressources as manuals in printed and electronic form, tutorials and on-line help are the access key to the software. While experienced users often need just an index of the available functions, novices and students require more assistance in the form of tutorials and sufficient manuals. These should also provide substantial help on the theoretical background of the available methods, since, as Tukey (1965) stated: "Most uses of the classical tools of statistics have been, are, and will be, made by those who know not what they do.".

3 Why do we need vertical integration?

Vertical integration means the smooth integration of statistical computing frameworks into completely different environments such as web browsers, electronic or printed books and standard application software. Table 2 depicts three examples, how this vertical integration may look like in practice: In scenario 1 Microsoft Excel reads data from a file or database and hands them over to an embedded computing module, for the graphical presentation of the results internal Excel routines are used. Scenario 2 shows how modern web standards as XML and SOAP may be used in a vertically integrated environment. Imagine a financial service provider such as Thomson Datastream or Bloomberg providing daily option data as *webservice* (`http://www.w3.org/TR/wsdl`), embedding the information in so called XML envelopes. A computing engine retrieves these data and calculates the desired statistics to provide for example a trading signal for further action. The last scenario uses special commands in LaTeX-source to generate output formats as HTML and PDF with embedded links to interactive examples that are run inside a webbrowser or a Java applet.

	Scenario 1	Scenario 2	Scenario 3
Organisation	Excel	webservice data	LaTeX
Analysis	computing engine	computing engine	Applet
Presentation	Excel	trading signal	webbrowser

Table 2: Three scenarios of vertically integrated software

Nowadays statistical software environments are of monolithic character, offering organization, analysis and presentation under one roof. Often these packages or significant parts of them have been written decades ago. As mentioned in Theus (1998), the basic graphics routines in *S* celebrated their 30th birthday in 1998. Reimplementation would certainly be necessary but seems impossible for one of the following reasons:

- The original programmers are not available anymore and left little or no information on the implementation details because no documentation standards had been defined.
- A growing codebase and pool of methods created interdependencies between e.g. computing engine and graphical user interface, so changing one part of the software may affect other parts as well.
- The original design approach did not allow modern extensions, their more rough than ready implementation causes problems with performance.
- Retaining compatibility to previous releases may force the developers to keep outdated code.

Vertically integrable software helps us to solve these problems. When all vital parts of the software framework are designed from scratch to work together smoothly via standardized interfaces, the modification of single parts is for sure much easier than working in a monolithic environment.

But even when the software is modular there are problems arising. Statistical frameworks using e.g. client-server architecture largely depend on the underlying communication protocols.

Figure 1 shows MD*Crypt, the TCP/IP based communication architecture used by the *XploRe Quantlet Server* to exchange information with its clients. While TCP/IP has the advantage of being available for all major platforms and allowing a reliable and relatively easy way of communication,

there is a drawback in speed due to the standard TCP/IP especially when the server application runs on the same machine as the client. Using technologies such as shared memory usually bring here a dramatic increase in performance.

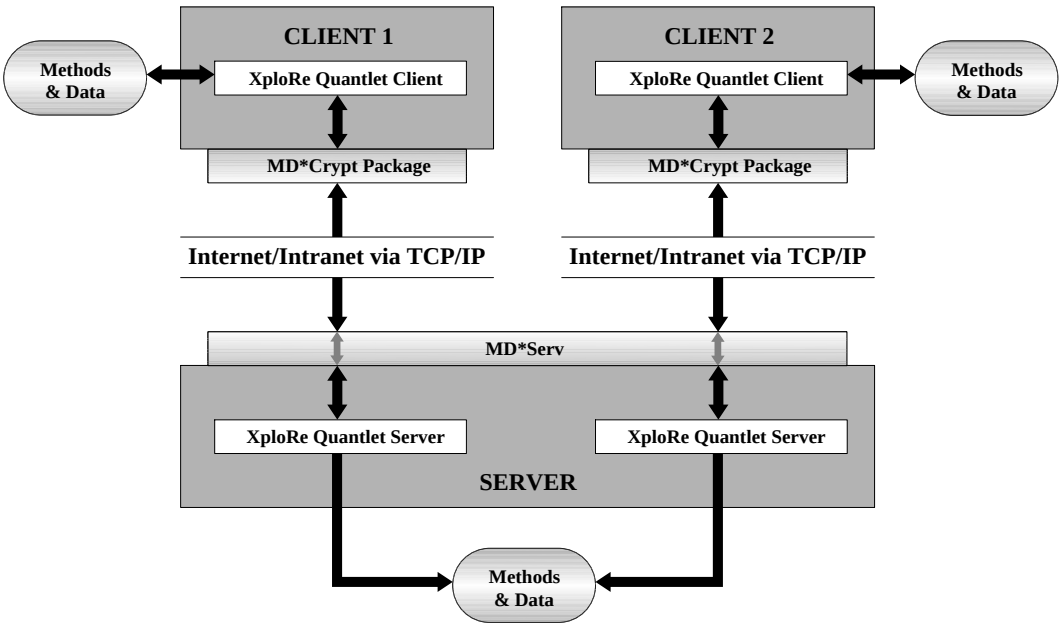


Figure 1: The MD*Crypt communication structure (Lehmann, 2003)

This problem of speed in communication might be improved by vertical integration. By offering different communication protocols or interfaces each user can select the best appropriate trade-off between speed and user friendlyness.

Last but not least the user interfaces offer opportunities for vertically integrated software. Essential 2 from Chambers and Lang (1999) suggests to offer several user interface instead of a single one. A monolithic software with a hard-wired GUI cannot satisfy this requirement. So a vertical separation of computational and presentation part is needed.

The question of user interfaces becomes especially interesting, when we take usability into account. There are different definitions for the term *usability*, a common one is given in Nielsen (2003), where usability is defined as "the measure of the quality of the user experience when interacting with something – whether a web site, a traditional software application, or any other device the user can operate in some way or another".

Nielsen (1993) divides the term usability into five components:

Learnability How easy can first-time users accomplish basic tasks?

Efficiency How quickly can users perform tasks once they know how to use the software?

Memorability When users return to the design after a period of not using it, how easily can they reestablish proficiency?

Errors How many (severe) errors do users make, how is the prevention of errors supported?

Satisfaction How pleasant is it to use the software?

Many software packages were originally written to satisfy the targeted needs of either the programmer himself or a number of experienced and skilled professionals, so usability concerns were rarely taken into consideration. Since the creators of the software knew how to interact with their own piece of work, they usually did not care how non-expert users would perceive the interface offered by the software.

Research in usability and standardized test procedures for usability as done by Jakob Nielsen (`http://www.useit.com`) are relatively unusual even today, so the "Hall of Shame" in interface design (Theus, 1999) is full of examples how not to design a user interface.

For answering the question, how a good interface design might look like, the "golden rules" of interface design (Shneiderman, 1997) may be taken as a guideline:

1. *consistency* The user expects similar reactions from the software in similar situations. When programmers work together in a team and no guidelines are given, each of the programmers implements his idea of a suitable reaction to actions from the user.
2. *shortcuts and feedback* While a beginner may need guidance in usage, the experienced users usuallys want to finish their task as quickly as possible. New users are mostly grateful for receiving feedback from the software, when certain tasks are finished but the professional user does not need this information.
3. *closed actions* Actions to be executed by the user should have well-defined start and end points.
4. *error handling* Error messages should be as short *and* informative as possible, "There was an error" is not sufficient.
5. *loss of control* Users prefer to act, not to react when working with software. Pure reacting may be perceived as loss of control.
6. *limited short term memory* Humans are able to store just a few items in their short term memory (Miller, 1956). The user interface should prevent the user from having to memorize different settings.

4 The Yxilon project

4.1 XploRe

Since the design of Yxilon is largely influenced by XploRe, we would like to give a short overview. XploRe was developed jointly by Humboldt-Universität zu Berlin and MD*Tech, a German software company, with the aim to provide a general purpose statistical computing environment for the quantitative analysis of data. XploRe quantlets cover a wide spectrum of statistical methods as Generalized (Partial) Linear Models, nonparametric methods (kernel estimation), single index and generalized additive models, ANOVA, etc. A focus lies on financial engineering functions for option pricing, Value-at-Risk and hedging strategies.

XploRe features a matrix-oriented programming language with C-style syntax and most of the internal functions are written in the XploRe language. Figure 2 depicts an example for the plotting of *Gamma* in the XploRe language, interested readers may refer to Härdle, Klinke and Müller (2000). External procedures written in C/C++ or Fortran may be executed from within XploRe via DLL

```
proc()=SFEgamma()
; beginning of procedure SFEgamma
         s=100                  ; stock price
         k=100                  ; exercise price
         r=0                    ; interest rate
         v=0.25                 ; volatility
         tau=0.5                ; time to maturity
         q=0                    ; dividend rate

         zeichen1="First variable, lower bound:"|"upper bound:"
         zeichen2="Second variable, lower bound:"|"upper bound:"
         values=50|150 ; predefined values first inputbox
         ss=readvalue(zeichen1,values)
         s1=ss[1]
         s2=ss[2]
         sw1=(s2-s1)/30

         values=0.05|1.0 ; predefined values second inputbox
         ss=readvalue(zeichen2,values)
         t1=ss[1]
         t2=ss[2]
         sw2=(t2-t1)/30
         lauf=grid(#(s1,t1),#(sw1,sw2),#(31,31))
         tau=lauf[,2]
         s=lauf[,1]

         d1=(log(s./k)+(r-q+v^2/2).*tau)/(v.*sqrt(tau))
         opv=(exp(-q.*tau).*pdfn(d1))./(s.*(v.*sqrt(tau)))
         dat= lauf~opv
         gs = grsurface(lauf~opv)
         plot3d(1,gs)
endp
; end of the procedure

library("xplore") ; load libraries
library("plot")
setsize(600,450) ; set display size
SFEgamma() ; call procedure
setgopt(plot3disp,1,1,"title", "Gamma","border",0) ; change layout of plot
```

Figure 2: XploRe code to plot the Gamma of a Call option
http://www.quantlet.com/mdstat/codes/sfm/SFMgamma.html

Key	effect
ExecuteProgram	loads XploRe code ("quantlets") from a file but does not show its sourcecode to the user
OpenInEditor	opens the quantlet in the built-in editor and allows modification by the user
ShowCommandWindow	defines, whether the command window is displayed that allows direct input of XploRe commands
ShowOutputWindow	The output window for textual output can be suppressed as well, this has proven useful when only a graphics is desired as output.

Table 3: Keys in XQC configuration file with their effect

function calls. An extensible HTML-based help system (Klinke and Witzel, 2002) offers detailed descriptions of all built-in functions.

XploRe has been implemented for Microsoft Windows and UNIX-based operating systems as Solaris and Linux in several versions as stand-alone, batch and Client-server version, a demo version may be downloaded from `http://www.xplore-stat.de`. A strong focus has been put on the scalability for different purposes, examples are the *XploRe Quantlet Client* (Lehmann, 2004) and *MD*ReX* (Aydınlı, Härdle and Neuwirth, 2003), an Add-In for Microsoft Excel.

4.2 The XploRe Quantlet Client

The XploRe Quantlet Client is a Java-based software that, besides running as an application, also runs as Applet from any Java-enabled webbrowser on any hardware platform supporting the Sun Microsystems JAVA framework.

The main goal in the development of XQC was to support teaching in several ways: it is used in projects for undergraduate students as MM*STAT (Müller, Rönz and Ziegenhagen, 2000), printed e-books as Härdle and Simar (2003) and Härdle, Franke and Hafner (2004) or in a variety of electronically published books (`http://www.xplore-stat.de/ebooks/ebooks.html`).

For this purpose the XQC implements the idea, also proposed by Chambers and Lang (1999), to offer different user interfaces to different groups of users. Via configuration file the XQC can be restricted to show only a subset of its features, for details see Table 3. This feature was implemented since different types of users have different needs. One of the main interests of undergraduates in statistics is to see changes in results when parameters are made. A standard example here is the role of the binwidth for histograms. They are mostly not interested in the implementation details while advanced students may also want to explore the sourcecode to use it in their own data analysis tasks.

The integration of interactive examples can be made either manually by changing the respective HTML-pages by hand or automatically be inserting certain commands into the LaTeX-source of a book or script. Using this MD*Book technology (Klinke and Lehmann, 2003) different output

formats as Postscript, PDF, HTML and a special HTML version enriched with Javascript can be created from one LaTeX-source.

The graphical user environment of XQC resembles the Windows stand-alone version of XploRe and most graphical functions of the stand-alone solution are supported. The XQC can be used or downloaded from http://www.xplore-stat.de.

4.3 MD*ReX

The MD*ReX framework (Aydınlı, Härdle and Neuwirth, 2003) uses the *Common Object Model*, a standardized architecture developed by Microsoft. The COM technology allows objects to communicate with each other regardless of which language they are written in or on which machine they are located. Since COM is part of all available Windows versions and integral part of the Microsoft office applications, it allows a smooth integration into these application. From a technical point of view MD*ReX serves as in-process COM Server and as the XploRe Quantlet Client it uses the MD*Crypt protocol as communication layer.

Figure 5 shows a screenshot of MD*ReX embedded in Excel. By the additional toolbar the user can connect to local or remote XploRe Quantlet Servers and store/retrieve data. The advantage of MD*ReX is the smooth integration into Excel, only little additional knowledge is needed to work with the Excel-MD*ReX combination.

4.4 Why Yxilon?

During the development of XploRe we faced several of the problems mentioned above. To keep and expand our expertise in statistical software we decided to form the Yxilon project. With the only premiss to be compatible with existing quantlets, we decided to change the major parts of the XploRe structure, in particular concerning:

- a strict separation of (G)UI and computing kernel, communication via light-weight protocol
- the demerger of data structures and reduction of kernel functionality
- a package and documentation system focussing on standard web techniques as ZIP and XML
- published under Free-BSD license (http://www.quantlet.org)

Before we examine some of these points in detail we can get an overview of the Yxilon architecture from Figure 6. The Yxilon core is formed of:

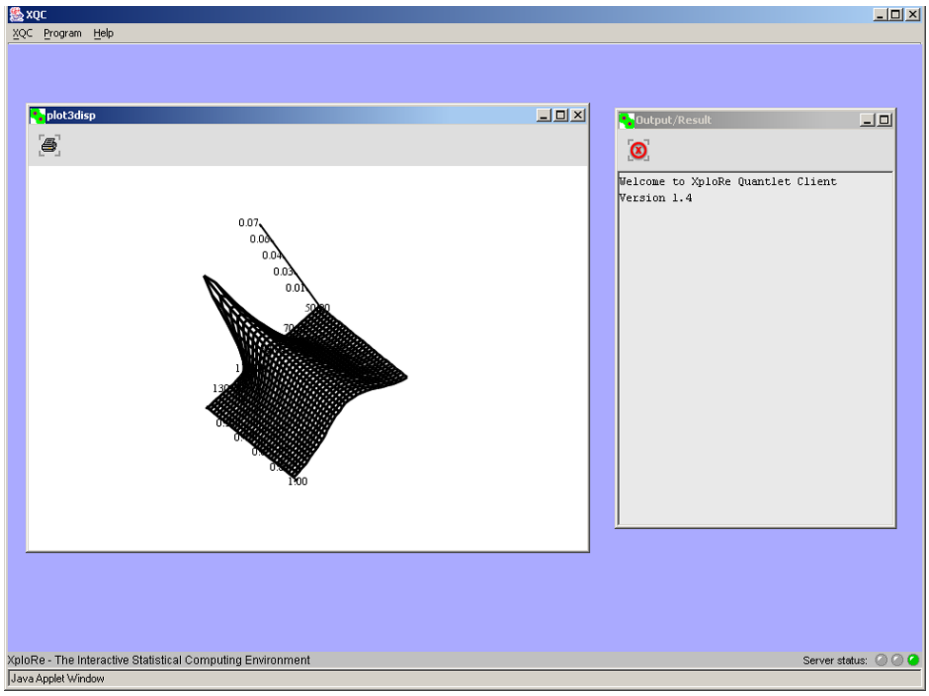
Object database: storing the data objects (lists, matrices, quantlets) for further usage

Parser: analysing the quantlet code, either generating C++/Java source or calling the interpreter

Interpreter: executing quantlet code directly

Database import/export filters: technically these import and export filters are clients without an own user interface

Clients: the other group besides the database plug-ins, including non-graphical and graphical user clients



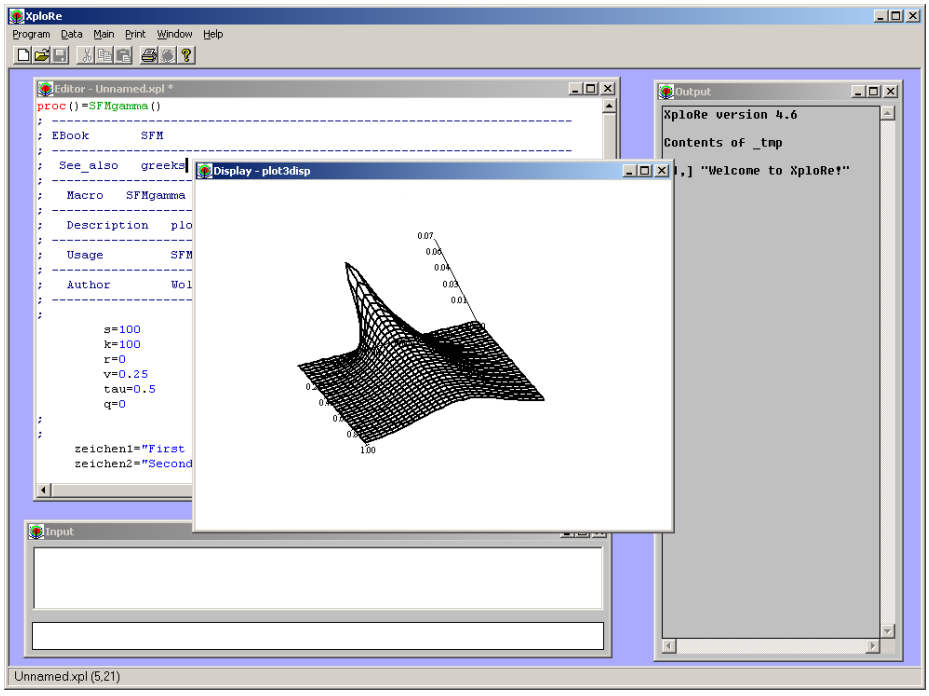


Figure 3: The output of SFEgamma.xpl (Figure 2) in XQC (execute) and Windows stand-alone version

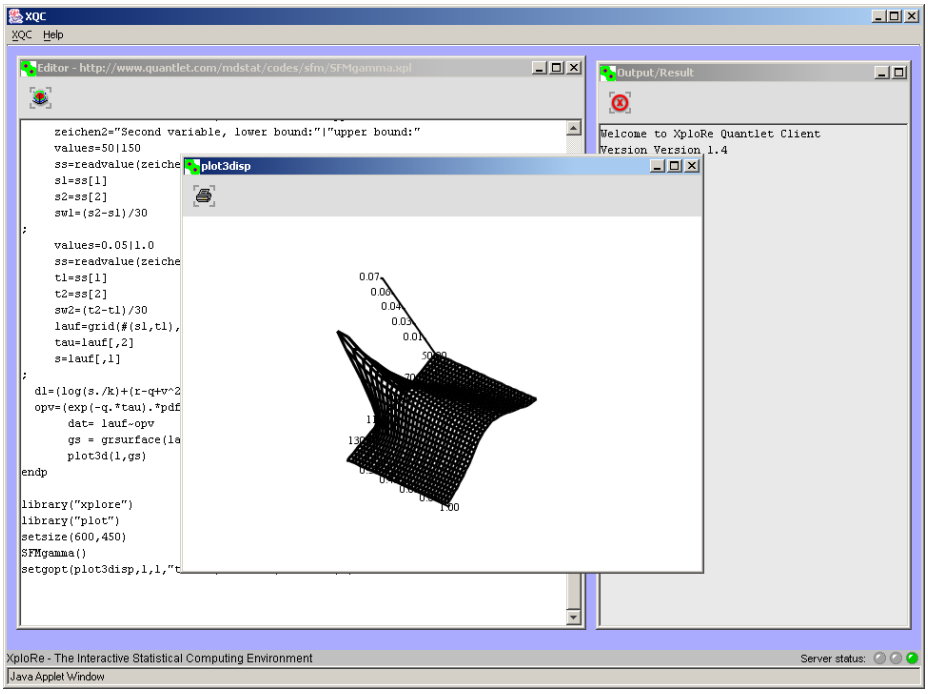


Figure 4: The output of SFEgamma.xpl in the XQC edit-version

4.5 Separation of Computing and Interface Components

The separation of computing engine and visualisation, that was already started with the client-server system of XploRe Quantlet Server and Client, is the basis for the communication of all clients and the computing kernel in Yxilon.

Compiled into the database component or situated directly before will be the server component dispatching the information flow from and to the different clients. As mentioned above the binary socket communication via the proprietary communication protocol has proven to be relatively slow since an unavoidable TCP/IP overhead has to be used for each data package. The conclusion is that at least for local environments, where client and kernel run on one machine, faster methods as shared memory access have to be implemented. This solution has also been used by JStatCom (Krätzig, 2004), a framework for econometric routines.

For remote connections the existing MD*Crypt protocol has to be evaluated and compared with competing technologies as RPC, RMI and CORBA:

RPC Remote Procedure Calls, developed by SUN Microsystems and language/processor-independent. Can only use native datatypes that have to be converted for different machines. Has the disadvantag of not being supported by Java natively and requiring more overhead than RMI.

RMI Remote Method Invocation, a relatively simple solution to access remote functions, but mostly limited to Java.

CORBA Common Object Request Broker Architecture, a more complex framework compared with RMI, implemented for different programming languages on several architectures

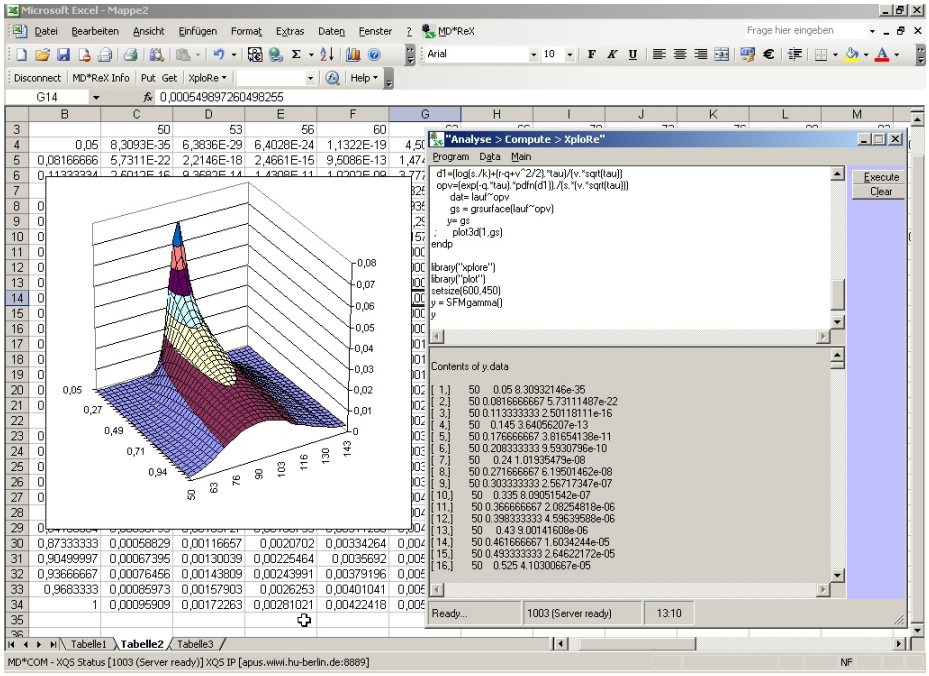


Figure 5: MD*ReX running with Excel 2003

SOAP One of the newer language independent protocols, using established web standards as HTTP and XML. The client and the server transmit parameters and results via XML, this holds the disadvantage of SOAP as well: the representation in XML may even triple the amount of data that has to be transferred, on both sides the XML has to be evaluated.

4.6 Compilation versus Interpretation

While XploRe is a completely interpreted language, Yxilon will use in the first step of the implementation compiled code. The quantlet code that is provided by clients is parsed and a parse tree is set up. On the basis of this generated tree, a 'treewalker' routine generates Java respective C++ sourcecode which is stored on the harddisk. The client afterwards calls an installed JAVA or C++ compiler that creates the native code or Java bytecode.

Important is here, also concerning usability, that the compilation process is hidden from the user. There must be no difference compared with an interpreting solution as it was provided by XploRe.

In the second step an interpreter will be written, since for small programming tasks the write-compile-test-recompile cycle is inefficient.

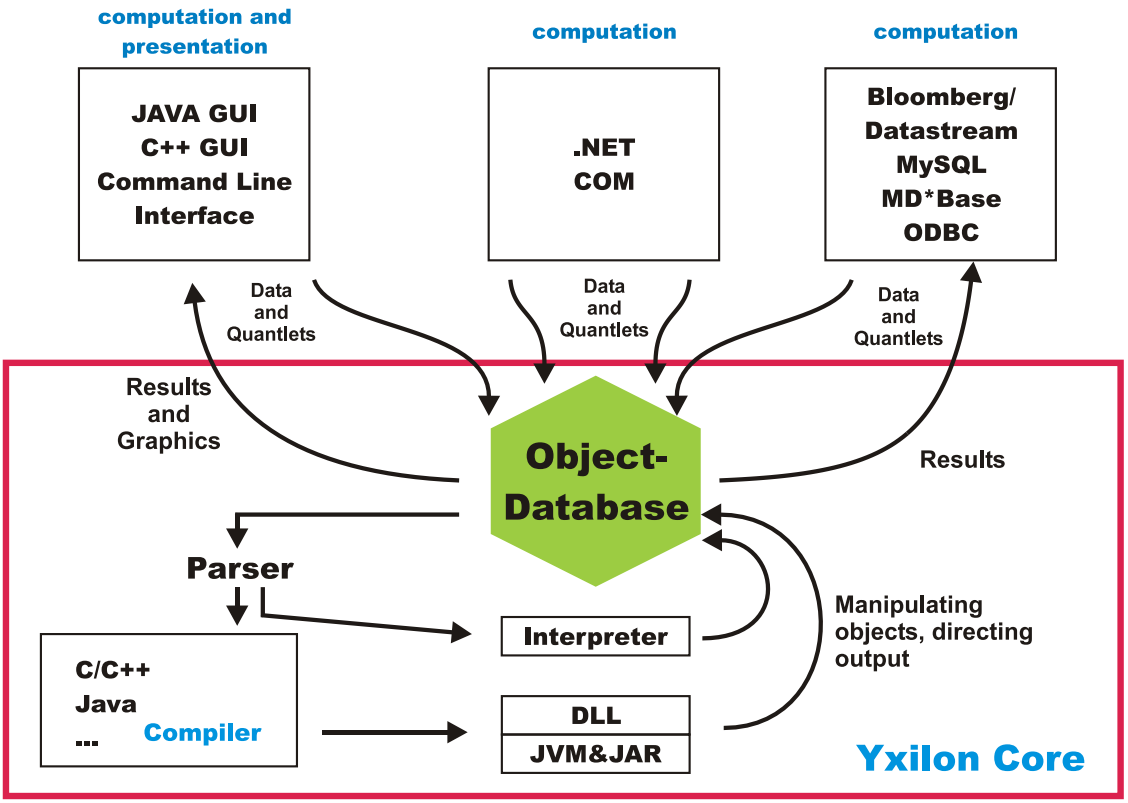


Figure 6: internal structure of Yxilon

4.7 Demerging Data Structures and Reduction of Kernel Functionality

The data storage unit in XploRe was tied closely to the parsing and interpreter parts of the kernel. The final goal in Yxilon is to have the database running non-stop as service or demon with clients connecting to it. In between we implement a solution that allows the clients to create own instances of the object database to store the necessary data.

Each software needs to have specific data structures to store internal settings as paths or the size of output graphics. In principle there are two methods to access these datastructures, directly via setting flags or values as done by GAUSS or indirectly via commands that change the values.

XploRe for example uses two commands to modify internal settings: `getenv` retrieves the current settings and `setenv` stores new values. In Yxilon we want to modify these commands and rather use global lists to store these values instead.

Further measures to reduce the number of commands the core needs to hold are the merger of commands and the outsourcing to DLL files. To add elements to a list XploRe uses `append`, to add elements to a vector $\sim$ and | and to concatenate strings the `+` operator.

The complete restructuring process includes 52% of all internal commands to be outsourced to dll-files and 24% changed from a command-oriented access to direct access structures.

5 Summary

Yxilon is our proposal to answer the question of how future statistical packages may look like. We will fulfill the essentials mentioned above in the following way:

Multiple front-ends: There will be two groups of frontends, user-oriented as graphical and non-graphical user interfaces as well as modules that can be vertically integrated in other standard software environments.

Different GUIs: A Java GUI in the first step, a C++ version to follow.

Extensibilty: Implicitly given since Yxilon is a full programming language.

Internet abilities: Reading and saving quantlets and data from remote locations via standard protocols.

Database support: Database support in form of DLL and JAR files.

Interactive graphics: Interactive graphics will be handled individually by each GUI.

MP Support: Thread-based models at least for the JAVA code generation

Optimization for Performance: Usage of compiled code instead of interpreation

Set of Methods: The existings XploRe methods will be usable from Yxilon.

Multiple language support: Via configuration file all captions in user interfaces can be changed.

We plan to have two stages in the development process, step one includes the completion of a parser, code generators for Java and C++ and a Java-based GUI. Furthermore the interface definitions for importing and exporting data have to be made in this step. The second step includes the implementation of the database as service or demon as well as the development of an on C++ based graphical user interfaces.

Details and downloads of the project are updated regularly and can be found at http://www.quantlet.org. A first impression on the parsing and code generation components of Yxilon can be found at http://141.20.100.252/yxilon-j/yxilon-j.html. We are looking for feedback from users and programmers and invite them to join us in this project.

References

Aydınlı, G., Härdle, W. and Neuwirth, E. (2003), Efficient and Secure Statistics in Office Applications, *in* M. Minnotte, J. Symanzik and E. Wegman, eds, 'Proc. of the 35th Symposium on the Interface "Security and Infrastructure Protection"'.

Becker, R. A. (1989), Statistical computing environments: Past, present and future, *in* '150 Years ASA - Sesquicentennial Invited Papers Sessions.', American Statistical Association.

Becker, R. A. (1994), 'A brief history of S', *Computational Statistics* pp. 81–110.

Chambers, J. M. (1999), 'Computing with data: "concepts and challenges"', *The American Statistician* .

Chambers, J. M. (2000), 'Users, programmers, and statistical software', *ASA Journal of Comp. and Graph.* .

Chambers, J. M. and Lang, D. T. (1999), Omegahat – a component-based statistical computing environment, *in* 'Proceedings of the 52nd ISI Session', ISI International Statistical Institute.

Chambers, J. M., Lang, D. T., James, D. and Hansen, M. (1998), Distributed computing with data: A corba-based approach, *in* '30th Symposium on the Interface', Interface Foundation of North America.

Härdle, W., Franke, J. and Hafner, C. (2004), *Einführung in die Statistik der Finanzmärkte*, 2nd edn, Springer.

Härdle, W., Klinke, S. and Müller, M. (2000), *XploRe Learning Guide*, Springer.

Härdle, W. and Simar, L. (2003), *Applied Multivariate Statistical Analysis*, Springer.

Klinke, S. (2004), Statistical user interfaces, *in* J. Gentle, W. Härdle and Y. Mori, eds, 'Handbook of Computational Statistics', Springer.

Klinke, S. and Lehmann, H. (2003), 'MD*Book and XQC – an architecture for reproducible research', SFB 373 Research Paper.
URL: *http://sfb.wiwi.hu-berlin.de*

Klinke, S. and Witzel, R. (2002), MD*Book online – a tool for creating interactive documents, *in* W. Härdle and B. Rönz, eds, 'Proceedings in Computational Statistics', Springer, pp. 449–454.

Krätzig, M. (2004), 'Creating user interfaces for econometric routines with JStatCom: An example for Ox', to be presented on the 2nd Oxmetrics User Conference, August 2004.

Lehmann, H. (2003), 'XploRe Quantlet Client – web service for mathematical and statistical computing', SFB 373 Research Paper.
URL: *http://sfb.wiwi.hu-berlin.de*

Lehmann, H. (2004), Client/Server based statistical computing, Dissertation, Humboldt-Universität zu Berlin.

McCullough, B. and Wilson, B. (1999), 'On the accuracy of statistical procedures in Microsoft Excel 97', *Computational Statistics & Data Analysis* **1**(31), 27–39.

Miller, G. A. (1956), 'The magical number seven, plus or minus two: Some limits on our capacity for processing information', *Psychological Review* .

Müller, M., Rönz, B. and Ziegenhagen, U. (2000), The multimedia project MM*Stat for teaching statistics, *in* J. Bethlehem and P. van der Heijden, eds, 'Proceedings in Computational Statistics'.

Nielsen, J. (1993), *Usability Engineering*, AP Professional.

Nielsen, J. (2003), 'Usability 101', Jakob Nielsen's Alertbox.
URL: *http://www.useit.com/alertbox/20030825.html*

Sawitzki, G. (1996), New directions in programming environments, *in* B. L. and N. Fisher, eds, 'Proceedings of the 28th Symposium on the Interface', Interface Foundation of North America.

Shneiderman, B. (1997), *Designing the User Interface*, 3. edn, Addison-Wesley Longman.

Theus, M. (1998), Java - the next generation of statistical computing?, *in* 'Proceedings of the 30th Symposium on the Interface'.

Theus, M. (1999), User interfaces of interactive statistical graphics software, *in* 'Proceedings of the 31th Symposium on the Interface'.

Tukey, J. W. (1965), 'The technical tools of statistics', *American Statistician* .